PL/I-80 V1.3 COMPILATION OF: INITDIR

D: Disk Print
N: Nesting Level Display

```
 1 b      initdir: procedure options(main);
 2 b
 3 b      declare
 4 b          cpm3              char(2) static initial('30');
 5 b
 6 b      /* fixed bug in clearout, buildnew, and reconstruction 11/12/82 */
 7 b
 8 b      /*
 9 b                copyright(c) 1982
10 b                digital research
11 b                box 579
12 b                pacific grove, ca
13 b                93950
14 b        */
15 b
16 b      /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
17 b
18 b
19 b                        * * *  DISK INTERFACE  * * *
20 b
21 b
22 b       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
23 b
24+b
25+b         dcl
26+b             memptr entry                 returns (ptr),
27+b             memsiz entry                 returns (fixed(15)),
28+b             memwds entry                 returns (fixed(15)),
29+b             dfcb0  entry                 returns (ptr),
30+b             dfcb1  entry                 returns (ptr),
31+b             dbuff  entry                 returns (ptr),
32+b             reboot entry,
33+b             rdcon  entry                 returns (char(1)),
34+b             wrcon  entry    (char(1)),
35+b             rdrdr  entry                 returns (char(1)),
36+b             wrpun  entry    (char(1)),
37+b             wrlst  entry    (char(1)),
38+b             coninp entry                 returns (char(1)),
39+b             conout entry    (char(1)),
40+b             rdstat entry                 returns (bit(1)),
41+b             getio  entry                 returns (bit(8)),
42+b             setio  entry    (bit(8)),
43+b             wrstr  entry    (ptr),
44+b             rdbuf  entry    (ptr),
45+b             break  entry                 returns (bit(1)),
46+b             vers   entry                 returns (bit(16)),
47+b             reset  entry,
48+b             select entry    (fixed(7))   returns (bit(16)),
49+b             open   entry    (ptr)        returns (bit(16)),
50+b             close  entry    (ptr)        returns (bit(16)),
51+b             sear   entry    (ptr)        returns (bit(16)),
52+b             searn  entry                 returns (bit(16)),
53+b             delete entry    (ptr)        returns (bit(16)),
54+b             rdseq  entry    (ptr)        returns (bit(16)),
55+b             wrseq  entry    (ptr)        returns (bit(16)),
56+b             make   entry    (ptr)        returns (bit(16)),
```

CP/M plus Ver 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
57+b            rename entry    (ptr)       returns (bit(16)),
58+b            logvec entry                returns (bit(16)),
59+b            curdsk entry                returns (fixed(7)),
60+b            setdma entry    (ptr),
61+b            allvec entry                returns (ptr),
62+b            ?disk entry,
63+b            rovec  entry                returns (bit(16)),
64+b            filatt entry    (ptr),
65+b            getdpb entry                returns (ptr),
66+b            getusr entry                returns (fixed(7)),
67+b            setusr entry    (fixed(7)),
68+b            rdran  entry    (ptr)       returns (bit(16)),
69+b            wrran  entry    (ptr)       returns (bit(16)),
70+b            filsiz entry    (ptr),
71+b            setrec entry    (ptr),
72+b            resdrv entry    (bit(16))   returns (bit(16)),
73+b            wrranz entry    (ptr)       returns (bit(16)),
74+b            testwr entry    (ptr)       returns (bit(16)),
75+b            lock   entry    (ptr)       returns (fixed(7)),
76+b            unlock entry    (ptr)       returns (fixed(7)),
77+b            multis entry    (fixed(7))  returns (fixed(7)),
78+b            ?mode entry     (bit(1)),
79+b            freesp entry    (fixed(7))  returns (bit(16)),
80+b            chain  entry                returns (bit(16)),
81+b            flush  entry                returns (fixed(7)),
82+b            setlbl entry    (ptr)       returns (bit(16)),
83+b            getlbl entry    (fixed(7))  returns (bit(8)),
84+b            rdxfcb entry    (ptr)       returns (bit(16)),
85+b            wrxfcb entry    (ptr)       returns (bit(16)),
86+b            settod entry    (ptr),
87+b            gettod entry    (ptr),
88+b            dfpswd entry    (ptr),
89+b            sgscb  entry    (ptr) returns(bit(8));
90 b
91 b
92+b         declare
93+b             seldsk  entry   (fixed(7)) returns(ptr),
94+b             settrk  entry   (fixed(15)),
95+b             setsec  entry   (fixed(15)),
96+b             rdsec   entry   returns(fixed(7)),
97+b             wrsec   entry   (fixed(7)) returns(fixed(7)),
98+b             sectrn  entry   (fixed(15), ptr) returns(fixed(15)),
99+b             bstdma  entry   (ptr);
100 b
101 b            %replace
102 b               TRUE          by '1'b,
103 b               FALSE         by '0'b;
104 b
105 b                                       /* directory array 4K */
106 b            declare
107 b                   1 dir_fcb(0:127),
108 b                    3 user          bit(8),
109 b                    3 rest(31)      char(1),
110 b
111 b                   1 outbuf(0:127),
112 b                    2 user          fixed(7),
113 b                    2 rest(31)      char(1),
114 b
115 b                   1 buffer2(0:127),
116 b                    2 user          bit(8),
117 b                    2 rest(31)      bit(8),
118 b
119 b                   1 outb(0:127)   based(outptr),
120 b                    2 rest          char(32),
121 b
122 b                   1 outb2(0:127)  based(outptr),
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
123 b                   2 user          bit(8),
124 b                   2 rest(31)      char(1),
125 b
126 b                  1 outb3(0:127)   based(outptr),
127 b                   2 user          fixed(7),
128 b                   2 rest(31)      bit(8),
129 b
130 b                  1 outb4(0:127)   based(outptr),
131 b                   2 sfcbm         char(1),
132 b                   2 sfcb(3),
133 b                    3 stamps       char(8),
134 b                    3 mode         bit(8),
135 b                    3 rest         char(1),
136 b                   2 frest         char(1),
137 b
138 b                  1 infcb(0:127)   based(dirptr),
139 b                   2 rest          char(32),
140 b
141 b                  1 infcb2(0:127)  based(dirptr),
142 b                   2 user          char(1),
143 b                   2 name          char(11),
144 b                   2 pmode         bit(8),
145 b                   2 junk1         char(11),
146 b                   2 stamp         char(8),
147 b
148 b                 1 clearbuf(0:127) based(clearptr),
149 b                   2 rest          char(32),
150 b
151 b                  zeroes(31)       bit(8) static init((31)'00000000'b);
152 b
153 b                                        /* directory array mask */
154 b           declare
155 b             1 dirm(0:127)       based(dirptr),
156 b               3 user           fixed(7),
157 b               3 fname          char(8),
158 b               3 ftype          char(3),
159 b               3 fext           bin fixed(7),
160 b               3 fs1            bit(8),
161 b               3 fs2            bit(8),
162 b               3 frc            fixed(7),
163 b               3 diskpass(8)    char(1),
164 b               3 rest           char(8);
165 b
166 b           declare                          /* disk parameter header mask */
167 b                  dphp            ptr,
168 b                  1 dph_mask      based(dphp),
169 b                   2 xlt1         ptr,
170 b                   2 space(9)     bit(8),
171 b                   2 mediaf       bit(8),
172 b                   2 dpbptr       ptr,
173 b                   2 csvptr       ptr,
174 b                   2 alvptr       ptr,
175 b                   2 dirbcb       ptr,
176 b                   2 dtabcb       ptr,
177 b                   2 hash         ptr,
178 b                   2 hbank        ptr,
179 b
180 b           xlt             ptr;    /* save the xlt ptr because of F10 buffer */
181 b
182 b           declare                  /* disk parameter block mask */
183 b              dpbp         ptr ext,
184 b              1 dpb_mask   based(dpbp),
185 b                2 spt      fixed(15),
186 b                2 blkshft  fixed(7),
187 b                2 blkmsk   fixed(7),
188 b                2 extmsk   fixed(7),
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
189 b                2 dsksiz  fixed(15),
190 b                2 dirmax  fixed(15),
191 b                2 diralv  bit(16),
192 b                2 checked fixed(15),
193 b                2 offset  fixed(15),
194 b                2 physhf  fixed(7),
195 b                2 phymsk  fixed(7),
196 b
197 b                   dspt    decimal(7,0),
198 b                   dblk    decimal(7,0);
199 b
200 b           declare
201 b              dir_blks(32)   bit(8),
202 b              errorcode      bit(16);
203 b
204 b           declare
205 b                   MAXSAVE         bin fixed(15),
206 b                   enddcnt         bin fixed(15),
207 b                   nxfcb           bin fixed(15),
208 b                   notsaved        bin fixed(15),
209 b                   xptr            pointer,
210 b
211 b                   1 XFCBs(1)      based(xptr),
212 b                    2 user         bin fixed(7),
213 b                    2 name         char(11),
214 b                    2 pmode        bit(8),
215 b                    2 stamp        char(8);
216 b
217 b
218 b     declare
219 b                   INITMSG         char(54) static initial
220 b                      ('INITDIR WILL ACTIVATE TIME STAMPS FOR SPECIFIED DRIVE.'),
221 b                   CONFIRM         char(60) varying static initial
222 b                      ('Do you want to re-format the directory on drive: '),
223 b
224 b                   ASKCLEAR        char(44) static initial
225 b                      ('Do you want the existing time stamps cleared'),
226 b                   RECOVER         char(50) varying static init
227 b                      ('Do you want to recover time/date directory space'),
228 b                   YN              char(10) static initial('  (Y/N)?  '),
229 b                   YES             char(1) static initial('Y'),
230 b                   lyes            char(1) static initial('y'),
231 b                   yesno           char(1),
232 b
233 b                   UPPERCASE       char(26) static initial
234 b                                           ('ABCDEFGHIJKLMNOPQRSTUVWXYZ'),
235 b                   LOWERCASE       char(26) static initial
236 b                                           ('abcdefghijklmnopqrstuvwxyz'),
237 b
238 b                   pass1           char(20) static initial
239 b                                           ('End of PASS 1.'),
240 b                   ERRORM          char(7) static initial('ERROR: '),
241 b                   TERM            char(30) static initial('INITDIR TERMINATED.'),
242 b                   errvers         char(30) static initial
243 b                                   ('Requires CP/M 3.0 or higher.'),
244 b                   errnotnew       char(31) static initial
245 b                                   ('Directory already re-formatted.'),
246 b                   errtoobig       char(30) static initial
247 b                                   ('Not enough room in directory.'),
248 b                   errpass         char(15) static initial('Wrong password.'),
249 b                   errSTRIP        char(30) varying static initial
250 b                                   ('No time stamps present.'),
251 b                   errMEM          char(30) varying static initial
252 b                                   ('Not enough available memory.'),
253 b                   errRO           char(20) varying static initial
254 b                                   ('Disk is READ ONLY.'),
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
255 b                   errWHAT         char(30) varying static initial
256 b                                   ('Cannot find last XFCB.'),
257 b                   errRSX          char(60) varying static initial
258 b                   ('Cannot re-format the directory with RSXs in memory.'),
259 b                   errunrec        char(19) static initial ('Unrecognized drive.'),
260 b
261 b                   errBIOS         char(20) static initial('Cannot select drive.');
262 b
263 b           declare
264 b                   outptr          pointer,
265 b                   bufptr1         pointer,
266 b                   bufptr2         pointer,
267 b                   dirptr          pointer,
268 b                   drivptr         pointer,
269 b                   clearptr        pointer,
270 b
271 b                   nempty          bin fixed(15),
272 b                   (nfcbs,nfcbs1)  bin fixed(15),
273 b                   lastsfcb        bin fixed(15),
274 b                   lastdcnt        bin fixed(15),
275 b                   (lasti,lastx)   bin fixed(15),
276 b                   lastsect        bin fixed(15),
277 b                   cleardcnt       bin fixed(15),
278 b                   (gsec,gtrk)     bin fixed(15),
279 b                   (dcnt,sect)     bin fixed(15),
280 b                   outdcnt         bin fixed(15),
281 b                   newdcnt         bin fixed(15),
282 b                   outidx          bin fixed(7),
283 b                   curdisk         bin fixed(7),
284 b                   newlasti        bin fixed(7),
285 b                   (sfcbidx,sfcboffs)  bin fixed(15),
286 b                   usernum         fixed(7),
287 b                   SFCBmark        fixed(7) static initial(33),
288 b                   Dlabel          bin fixed(7) static initial (32),
289 b
290 b                   Redo            bit(1),
291 b                   bad             bit(1),
292 b                   writeflag       bit(1),
293 b                   CLEARSECT       bit(1),
294 b                   CLEARSFCB       bit(1),
295 b                   labdone         bit(1) static initial(false),
296 b                   cversion        bit(16),
297 b                   READonly        bit(16),
298 b
299 b                   ptreos          pointer,
300 b                   EOS             bit(8) static initial('00'b4),
301 b                   CEOS            char(1) based (ptreos),
302 b
303 b                   fcb(32)         char(1),
304 b                   fcb0(50)        char(1) based (drivptr),
305 b                   dr0             fixed(7) based(drivptr),
306 b                   disks           char(16) static initial
307 b                                   ('ABCDEFGHIJKLMNOP'),
308 b                   drive           bin fixed(7),
309 b                   cdrive          char(1);
310 b
311 b     declare
312 b                   1 SCB,
313 b                    2 soffs        fixed(7),
314 b                    2 seter        fixed(7),
315 b                    2 value        char(2),
316 b
317 b                   ccppage         bit(8);
318 b
319 b     /*****************************************************************************
320 b
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
321 b
322 b                         ***  MAIN PROGRAM  ***
323 b
324 b
325 b     ****************************************************************************/
326 b
327 b           declare i       bin fixed(7);
328 b
329 b           cversion = vers();
330 b           if substr(cversion,9,8) < '31'b4 then call errprint((errvers));
331 b
332 b           soffs = 23;
333 b           seter = 0;
334 b           ccppage = sgscb(addr(SCB));             /* if RSX present then stop */
335 b           if substr(ccppage,7,1) = '1'b then call errprint(errRSX);
336 b
337 b           drivptr = dfcb0();                      /* get drive */
338 b           drive = dr0;
339 b           if dr0 > 16  then drive = 0;
340 b
341 c           do while(drive = 0);                    /* none recognized */
342 c                   call wrongdisk(i,drive);
343 c                   call getdisk(i,drive);
344 c           end;
345 b
346 b           cdrive = substr(disks,drive,1);
347 b
348 b           curdisk = curdsk();                     /* restore BIOS to this */
349 b
350 b           put edit(INITMSG,confirm,cdrive,YN)(skip(2),a,skip,a,a,a);
351 b           get list(yesno);
352 b           if yesno ^= YES & yesno ^= yes then call reboot;
353 b
354 b           READonly = revec();                     /* is the drive RO ? */
355 b           if substr(READonly,(17-drive),1) = '1'b then
356 b             call errprint(errRO);
357 b
358 b           call dselect(drive);
359 b           nfcbs = ((physmsk + 1)*4) - 1;          /* # fcbs/physical rcd - 1 */
360 b           nfcbs1 = nfcbs + 1;
361 b
362 b           dirptr = addr(dir_fcb(0));
363 b           dcnt = 0;
364 b           call read_sector(dcnt,dirptr);
365 b
366 b           call init;
367 b
368 b           call restore;
369 b
370 b     /****************************************************************************/
371 b
372 b
373 c     wrongdisk: procedure(i,drive);
374 c           declare (i,j,drive)     bin fixed(7);
375 c
376 c           put list(ERRORM,errunrec);
377 c           put skip list('DRIVE: ');
378 c                                                   /* print errant string */
379 c           j = i;
380 c           ptreos = addr(EOS);
381 d           do while(fcb0(j) ^= ' ' & fcb0(j) ^= EOS);
382 d                   put edit(fcb0(j))(a);
383 d                   j = j + 1;
384 d           end;
385 c           put skip;
386 c
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
387 c     end wrongdisk;
388 b
389 c     getdisk: procedure(i,drive);
390 c             declare (i,drive)       bin fixed(7);
391 c
392 c             put skip list('Enter Drive: ');
393 c             get list(fcb0(i));
394 c             fcb0(i) = translate(fcb0(i),UPPERCASE,LOWERCASE);
395 c             fcb0(i+1) = ':';
396 c
397 c             drive = index(disks,fcb0(i));
398 c
399 c     end getdisk;
400 b
401 b
402 b     /*************************************************************************/
403 b
404 b
405 c     init: procedure;
406 c
407 c             declare
408 c                     (i,j,k,l)       bin fixed(15);
409 c
410 c             call allxfcb;                   /* allocate XFCB data space */
411 c             call countdir;
412 c
413 c             lastx = nxfcb;
414 c             sect = sect - 1;
415 c             dcnt = dcnt - 1;                        /* reset to good dcnt */
416 c
417 d             if Redo then do;
418 d                     newdcnt = lastdcnt;
419 d                     newlasti = lasti;
420 d             end;
421 d             else do;
422 d                     newdcnt = lastdcnt + lastsfcb + (2 - mod(lastdcnt,3));
423 e                     if (newdcnt + 1) > dirmax then do;
424 e                             lastdcnt = lastdcnt - nempty;
425 e                             lastsfcb = lastdcnt/3 + 1;
426 e                             newdcnt = lastdcnt + lastsfcb + (2 - mod(lastdcnt,3));
427 e
428 e                             if (newdcnt + 1) > dirmax then
429 e                                call errprint(errtoobig);
430 e
431 e                             call collapse;          /* remove all empties by
432 e                                                        collapsing dir from top */
433 e                             lastsfcb = lastdcnt/3 + 1;
434 e                             newdcnt = lastdcnt + lastsfcb + (2 - mod(lastdcnt,3));
435 e                     end;
436 d                     newlasti = mod(newdcnt,nfcbs1) - 3 + mod(lastdcnt,3);
437 d             end;
438 c
439 c             outptr = addr(buffer2(0));              /* want to clear last read
440 c                                                        sector...buffer2 only used
441 c                                                        in collapse so it is free */
442 c             call clearout;
443 c             clearptr = outptr;
444 c             outptr = addr(outbuf(0));
445 c             call clearout;                          /* zero output buffer */
446 c
447 c
448 c     /*************************************************************************/
449 c
450 c
451 d             do while(lastsect < sect );             /* clear from end of dir */
452 d                     call write_sector(dcnt,outptr);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
453 d                   dcnt = dcnt - nfcbs1;
454 d                   sect = sect - 1;
455 d           end;
456 c
457 d           if (nempty - 1) ^= dirmax then do;      /* if there are files on dir */
458 d
459 d                                                   /* bottom of directory is
460 d                                                      now all E5 and 21...
461 d                                                      it is positioned to the
462 d                                                      last good sector of the old
463 d                                                      directory.  */
464 d              dcnt = lastdcnt;
465 d              enddcnt = newdcnt;
466 d              call read_sector(dcnt,dirptr);       /* read last good sector */
467 d
468 d              outidx = newlasti;                   /* index into out buffer */
469 d              call buildnew(lasti);                /* fill in outbuff from the
470 d                                                      bottom up...need this call
471 d                                                      because lasti may be in
472 d                                                      middle of read buffer */
473 e              do while(dcnt >= 0);
474 e                                                   /* as soon as we are finished
475 e                                                      with reading old sector,
476 e                                                      then go clear it.  This
477 e                                                      should limit possibility
478 e                                                      that duplicate FCB's occur.
479 e                                                   */
480 e                   call read_sector(dcnt,dirptr);
481 e                   call buildnew(nfcbs);
482 e              end;
483 d
484 d           end;                                    /* virgin dir */
485 c
486 c           else call write_sector(0,outptr);       /* write last sector */
487 c
488 d           do while(notsaved > 0);
489 d                   call moreXFCB;
490 d           end;
491 c
492 c     end init;
493 b
494 b     /*************************************************************************/
495 b
496 b
497 c     strip: procedure;
498 c
499 c                                   /* remove all SFCB from directory by jamming
500 c                                      E5 into user field.  Also turn off time/date
501 c                                      stamping in DIR LABEL.  */
502 c
503 c           declare (i,j)           bin fixed(7),
504 c                   1 direct(0:127) based(dirptr),
505 c                    2 junk1        char(12),
506 c                    2 ext          bit(8),
507 c                    2 rest         char(19),
508 c
509 c                   olddcnt         bin fixed(15);
510 c
511 c
512 c           dcnt = 0;
513 c
514 d           do while(dcnt <= dirmax);
515 d
516 d                   call read_sector(dcnt,dirptr);
517 d
518 d                   olddcnt = dcnt;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
519 e                   do i = 0 to nfcbs while(dcnt <= dirmax);
520 e
521 e                           if ^labdone then
522 f                               if dirm(i).user = Dlabel then do;
523 f                                   call getpass(i);
524 f                                   direct(i).ext = direct(i).ext & '10000001'b;
525 f                                   labdone = true;
526 f                               end;
527 e
528 e                           if dirm(i).user = SFCBmark then
529 e                               dir_fcb(i).user = 'E5'b4;
530 e
531 e                           dcnt = dcnt + 1;
532 e                   end;
533 d
534 d                   call write_sector(olddcnt,dirptr);
535 d           end;
536 c
537 c   end strip;
538 b
539 b
540 b   /*************************************************************************/
541 b
542 b
543 b
544 c   countdir: procedure;
545 c           declare i       bin fixed(7);
546 c
547 c                                       /* there are 5 valid sets of codes in
548 c                                          the user field:
549 c
550 c                                               E5      - empty
551 c                                               0-15    - user numbers
552 c                                               32      - Directory label
553 c                                               33      - SFCB marker
554 c                                               16-31   - XFCB marker
555 c
556 c                                          This routine counts the # of used
557 c                                          directory slots ignoring E5.
558 c                                          NOTE: if SFCB present then last
559 c                                                slot = SFCB */
560 c
561 c           Redo = false;
562 c           nempty = 0;
563 c           sect = 0;
564 c           nxfcb = 0;
565 c           notsaved = 0;
566 c           bad = true;
567 c                                       /* If dir is already time stamped then
568 c                                          SFCBs should appear in every sector,
569 c                                          notably the first sector. Thus,
570 c                                          test first sector.  If first sector
571 c                                          has SFCB then all do.  If none in
572 c                                          first & they appear later then
573 c                                          INITDIR was probably interrupted.
574 c                                          In that case, zap the found SFCB's
575 c                                          and treat dir as virgin.  */
576 c
577 c           if dirm(3).user = SFCBmark then bad = false;
578 c
579 d           do while(dcnt <= dirmax);
580 e                   do i = 0 to nfcbs while(dcnt <= dirmax);
581 f                           if dir_fcb(i).user ^= 'E5'b4 then do;
582 f                                   usernum = dirm(i).user;
583 f
584 f                                   if ^Redo & usernum = 33 then call query;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
585 f
586 f                                   if usernum > 15 & usernum < 32 then
587 f                                      call getXFCB(i);
588 f
589 f                                           /* if LABEL then check for password...
590 f                                              may terminate in getpass */
591 f
592 f                                   else if usernum = Dlabel then call getpass(i);
593 f
594 f                                   if (usernum < 33) | (^bad & usernum = 33) then
595 g                                          do;
596 g
597 g                                          lasti = i;
598 g                                          lastsect = sect;
599 g                                          lastdcnt = dcnt;
600 g                                   end;                  /* bad...*/
601 f                                   else if usernum = 33 then nempty = nempty + 1;
602 f
603 f                             end;                     /* E5 ... */
604 e                             else nempty = nempty + 1;
605 e                             dcnt = dcnt + 1;
606 e                       end;
607 d
608 d                       sect = sect + 1;
609 d                       call read_sector(dcnt,dirptr);
610 d                 end;
611 c
612 c                 if ^Redo then lastsfcb = lastdcnt/3 + 1;
613 c
614 c           end countdir;
615 b
616 c           getXFCB: procedure(i);
617 c                 declare i       bin fixed(7);
618 c
619 d                  if nxfcb <= MAXSAVE then do;
620 d                        nxfcb = nxfcb + 1;
621 d                        XFCBs(nxfcb).user = usernum - 16;
622 d                        XFCBs(nxfcb).name = infcb2(i).name;
623 d                        XFCBs(nxfcb).pmode = infcb2(i).pmode;
624 d                        XFCBs(nxfcb).stamp = infcb2(i).stamp;
625 d                  end;
626 c                  else notsaved = notsaved + 1;
627 c
628 c           end getXFCB;
629 b
630 b
631 c           allxfcb: procedure;
632 c
633 c                                   /* allocates largest available block of space
634 c                                      to be used in storing XFCB info.
635 c                                           maxwds & allwds use word units */
636 c
637 c                 declare maxwds          entry returns(fixed(15)),
638 c                         allwds          entry(fixed(15)) returns(pointer),
639 c                         size            bin fixed(15);
640 c
641 c                 size = maxwds();                /* get largest block in free space */
642 c                 if size <= 10 then call errprint(errMEM);
643 c
644 c                 xptr = allwds(size);            /* reserve it */
645 c                 MAXSAVE = (2*size)/21;          /* # XFCBs that can be saved */
646 c
647 c           end allxfcb;
648 b
649 b
650 c           query: procedure;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
651 c
652 c             if bad then return;
653 c
654 c             put skip(2) list(errnotnew);
655 c
656 c                                                  /* check to see if user wants
657 c                                                     to strip SFCB's */
658 d             if ^asker(RECOVER) then do;
659 d                     Redo = true;
660 d                     CLEARSFCB = false;
661 e                     if asker(ASKCLEAR) then do;
662 e                             CLEARSFCB = true;
663 e                             return;
664 e                     end;
665 d             end;
666 c             else call strip;                     /* this will end down here
667 c                                                     after stripping */
668 c
669 c             call restore;                        /* dir is already formattted &
670 c                                                     user does not want to clear
671 c                                                     old SFCB's....just stop */
672 c
673 c       end query;
674 b
675 c       buildnew: procedure(endidx);
676 c             declare (i,j,k,endidx)          bin fixed(15);
677 c
678 c       declare 1 ot(0:127)     based(outptr),
679 c                2 user         fixed(7),
680 c                2 fname        char(8),
681 c                2 ftype        char(3),
682 c                2 rest         char(20);
683 c
684 c                                          /* build output buffer from
685 c                                               input(end) to input(0).
686 c                                               K => refers to input */
687 c             K = endidx;
688 d             do while(K >= 0);
689 d                     usernum = dirm(k).user;
690 d
691 d                     outb(outidx).rest = infcb(K).rest;
692 d
693 e                     if usernum = SFCBmark then do;
694 e                             if bad then outb2(outidx).user = 'E5'b4;
695 e                             else if CLEARSFCB then outb3(outidx).rest = zeroes;
696 e                     end;
697 d
698 e                     if usernum < 16 then do;
699 e                        if nxfcb > 0 then         /* if fcb is ex=0 and XFCB
700 e                                                     exists then check for
701 e                                                     possible SFCB update */
702 e                             call putXFCB(K);
703 e                     end;
704 d
705 d                     if ^Redo & mod(outidx,4) = 0 then outidx = outidx - 2;
706 d                     else outidx = outidx - 1;
707 d
708 d                     K = K - 1;
709 d                     dcnt = dcnt - 1;
710 d
711 e                     if outidx < 0 then do;
712 e                             if dcnt > 14 then
713 e                                if mod(dcnt + 1,nfcbs1) = 0 then
714 e                                     call write_sector(dcnt + 1,clearptr);
715 e                             call write_sector(newdcnt,outptr);
716 e                             newdcnt = newdcnt - nfcbs1;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
717 e                           outidx = nfcbs - 1;
718 e                           if Redo then outidx = outidx + 1;
719 e                   end;
720 d           end;
721 c
722 c     end buildnew;
723 b
724 b
725 b     /*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
726 b
727 b
728 c     compare: procedure(k) returns(fixed(7));
729 c
730 c             declare (i,j,k)         bin fixed(7),
731 c                     1 direc(0:127)  based(dirptr),
732 c                      2 user         fixed(7),
733 c                      2 name(11)     char(1),
734 c                      2 rest         char(20),
735 c
736 c                     1 XFCB2(1)      based(xptr),
737 c                      2 user         char(1),
738 c                      2 name(11)     char(1),
739 c                      2 rest         char(9);
740 c
741 c                                             /* compare fcb with XFCB list;
742 c                                                return position in list if
743 c                                                found, 0 otherwise.
744 c                                                Nullify usernum field in
745 c                                                XFCB list (=99) if found.
746 c                                                Decrement #xfcb as well.*/
747 d             do i = 1 to nxfcb;
748 e                     if XFCBs(i).user ^= 99 then do;
749 f                             if XFCBs(i).user = direc(k).user then do;
750 f
751 g                                     do j = 1 to 11;
752 g                                        if direc(k).name(j) ^= XFCB2(i).name(j)
753 g                                           then go to outx;
754 g                                     end;
755 f
756 f                                                     /* found a match */
757 f                                     XFCBs(i).user = 99;
758 f                                     nxfcb = nxfcb - 1;
759 f                                     return(i);
760 f
761 f     outx:                           end;
762 e                     end;
763 d             end;
764 c
765 c             return(0);
766 c
767 c     end compare;
768 b
769 c     moreXFCB: procedure;
770 c                                     /* we could not store all the xfcb's in memory
771 c                                        available, so now must make another pass &
772 c                                        store as many XFCB as possible.
773 c                                        'notsaved' > 0 ==> we may have to
774 c                                                           do this again.  */
775 c             declare (i,k)   bin fixed(7);
776 c
777 c             dcnt = enddcnt;                 /* go to end of directory */
778 c             if ^findXFCB(k) then            /* work backwards trying to find
779 c                                                last known XFCB...if not found
780 c                                                then something very strange has
781 c                                                happened;
782 c                 call errprint(errWHAT);
```

COPYRIGHT © 1932
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
783 c
784 c             notsaved = 0;                   /* now in last sector where last XFCB
785 c                                                occurs...look for other XFCB that
786 c                                                we know is there.  */
787 c             nxfcb = 0;
788 c
789 c             dcnt = dcnt + 1;
790 c             lastdcnt = dcnt;                /* save position of last XFCB + 1 */
791 c             lasti = k + 1;                  /* index in sector */
792 d             do while(dcnt <= enddcnt);
793 e                     do i = k+1 to nfcbs while(dcnt <= enddcnt);
794 e                             usernum = dirm(i).user;
795 e                             if usernum > 15 & usernum < 32 then call getXFCB(i);
796 e                             dcnt = dcnt + 1;
797 e                     end;
798 d                     k = 0;
799 d                     call read_sector(dcnt,dirptr);
800 d             end;
801 c
802 c             dcnt = 0;                               /* go to start of dir */
803 d             do while(dcnt <= enddcnt);
804 d                     call read_sector(dcnt,dirptr);
805 d                     outdcnt = dcnt;
806 d                     writeflag = false;              /* putXFCB sets when it finds a
807 d                                                        match */
808 d
809 e                     do k = 0 to nfcbs while(dcnt <= enddcnt);
810 e                             outidx = k;
811 e                             if dirm(k).user < 16 then call putXFCB(k);
812 e                             dcnt = dcnt + 1;
813 e                     end;
814 d                     if writeflag then call write_sector(outdcnt,dirptr);
815 d             end;
816 c
817 c     end moreXFCB;
818 b
819 c     findXFCB: procedure(idx) returns(bit(1));
820 c
821 c                                     /* find the last known XFCB...starts from the
822 c                                        last written sector in the dir and goes
823 c                                        backwards...hopefully that's faster */
824 c             declare idx     fixed(7);
825 c
826 d             do while(dcnt > 0);
827 d                     call read_sector(dcnt,dirptr);
828 e                     do idx = 0 to nfcbs while(dcnt > 0);
829 e                             usernum = dirm(idx).user;
830 e                             if usernum > 15 & usernum < 32 then
831 e                                     if XFCBs(lastx).name = infcb2(idx).name then
832 e                                             return(true);
833 e                             dcnt = dcnt + 1;
834 e                     end;
835 d             end;
836 c
837 c             return(false);          /* big trouble...*/
838 c
839 c     end findXFCB;
840 b
841 b
842 c     putXFCB: procedure(k);
843 c                                     /* if this is extent 0 fold and names match
844 c                                        then update SFCB from XFCB */
845 c             declare (k,j)   fixed(7);
846 c
847 d                                     if dirm(k).fext <= dpb_mask.extmsk then do;
848 d                                        j = compare(k);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
349 e                               if j ^= 0 then do;
350 e
351 e                                                   /* fcb matches XFCB...
352 e                                                      update the SFCB */
353 e                                 sfcboffs = mod(outidx+1,4);
354 e                                 sfcbidx = outidx + (4 - sfcboffs);
355 e                                 outb4(sfcbidx).sfcb(sfcboffs).stamps =
356 e                                                           XFCBs(j).stamp;
357 e                                 outb4(sfcbidx).sfcb(sfcboffs).mode =
358 e                                                           XFCBs(j).pmode;
359 e                                 writeflag = true;
360 e                               end;
361 d                             end;                  /* extent 0 ? */
362 c
363 c     end putXFCB;
364 b
365 b
366 c     errprint: procedure(msg);
367 c             declare
368 c                     msg             char(60) varying;
369 c
370 c             put edit(ERRORM,msg,TERM)(skip(2),a,a,skip,a);
371 c             put skip(2);
372 c
373 c             call restore;
374 c
375 c     end errprint;
376 b
377 b
378 c     asker: procedure(msg) returns(bit(1));
379 c
380 c             declare msg             char(60) varying;
381 c
382 c             put skip list(msg,YN);
383 c             get list(yesno);
384 c
385 c             if yesno ^= YES & yesno ^= lyes then return(false);
386 c
387 c             return(true);
388 c
389 c     end asker;
390 b
391 b
392 c     clearout: procedure;
393 c             declare
394 c                     (i,j)   bin fixed(7);
395 c
396 d             do i = 0 to nfcbs;
397 d                     if mod(i+1,4) ^= 0 then outb2(i).user = 'E5'b4;
398 d                     else outb3(i).user = SFCBmark;
399 d
400 e                     do j = 1 to 31;
401 e                             outb3(i).rest(j) = '00000000'b;
402 e                     end;
403 d             end;
404 c
405 c     end clearout;
406 b
407 c     getpass: procedure(fcbx);
408 c                                       /* Drive may be password protected...
409 c                                          Get passw from user and compare
410 c                                          with Password in label.
411 c                                          Label password is encoded by first
412 c                                          reversing each char nibble and then
413 c                                          XOR'ing with the sum of the pass.
414 c                                          S2 in label = that sum.  */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
915 c
916 c             declare
917 c                     passwd(8)       bit(8) based(passptr),
918 c
919 c                     passptr         pointer,
920 c                     convptr         pointer,
921 c                     pchar(8)        bit(8),
922 c                     cvpass(8)       char(1) based(convptr),
923 c                     inpass          char(8),
924 c                     (i,j,fcbx)      bin fixed(7);
925 c
926 c             labdone = true;
927 c
928 c             passptr = addr(dirm(fcbx),diskpass);
929 c             convptr = addr(pchar(1));
930 c
931 d             do i = 1 to 8;                  /* XOR each character  */
932 d                     pchar(i) = bool(passwd(i),dirm(fcbx),fs1,'0110'b);
933 d             end;
934 c
935 c             if cvpass(8) <= ' ' then return; /* no password */
936 c
937 c             put skip(2) list('Directory is password protected.');
938 c             put skip list('Password, please.  >');
939 c             get list(inpass);
940 c             inpass = translate(inpass,UPPERCASE,LOWERCASE);
941 c
942 c             j = 8;
943 d             do i = 1 to 8;
944 d                     if substr(inpass,i,1) ^= cvpass(j) then call errprint(errpass);
945 d                     j = j - 1;
946 d             end;
947 c
948 c     end getpass;
949 b
950 c     collapse: procedure;
951 c
952 c             declare whichbuf        bin fixed(7),
953 c                     enddcnt         bin fixed(15),
954 c                     (i,nout1,nout2) bin fixed(7);
955 c
956 c             dcnt = 0;
957 c             sect = 0;
958 c             outdcnt = 0;
959 c             whichbuf = 0;
960 c             nout1 = 0;
961 c             nout2 = 0;
962 c             lastsect = 0;
963 c             enddcnt = lastdcnt + nempty;
964 c             lastdcnt = 0;
965 c             bufptr1 = addr(outbuf(0));
966 c             bufptr2 = addr(buffer2(0));
967 c
968 d             do while(dcnt <= enddcnt);              /* read up to last dcnt */
969 d
970 d                     call read_sector(dcnt,dirptr);
971 d
972 e                     do i = 0 to nfcbs while(dcnt <= enddcnt);
973 e                             if dir_fcb(i),user ^= 'E5'b4 &
974 f                               dirm(i),user ^= SFCBmark then do;
975 f
976 f                                 if whichbuf = 0 then
977 f                                         call fill(bufptr1,i,nout1,whichbuf);
978 f                                  else call fill(bufptr2,i,nout2,whichbuf);
979 f                             end;
980 e                             dcnt = dcnt + 1;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
981 e                   end;
982 d
983 d                   sect = sect + 1;
984 d                   if nout1 = nfcbs1 then call flush_write(nout1,bufptr1);
985 d                   else if nout2 = nfcbs1 then call flush_write(nout2,bufptr2);
986 d           end;
987 c
988 c           dcnt = dcnt - 1;                        /* fill unused slots in buffer
989 c                                                      with empty...scratch rest of
990 c                                                      dir */
991 c           if whichbuf = 0 then call fill2(bufptr1,nout1);
992 c           else call fill2(bufptr2,nout2);
993 c
994 c     end collapse;
995 b
996 c     fill: proc(bufptr,i,nout,whichbuf);
997 c           declare bufptr          pointer,
998 c                   (i,j,nout)      bin fixed(7),
999 c                   whichbuf        bin fixed(7),
1000 c
1001 c                  1 buffer(0:127) based(bufptr),
1002 c                   2 out          char(32);
1003 c
1004 c          buffer(nout).out = infcb(i).rest;
1005 c
1006 c          lastdcnt = lastdcnt + 1;
1007 c          nout = nout + 1;
1008 c          if nout = nfcbs1 then whichbuf = mod((whichbuf + 1),2);
1009 c
1010 c    end fill;
1011 b
1012 c    flush_write: proc(nout,bufptr);
1013 c          declare nout            bin fixed(7),
1014 c                  bufptr          pointer;
1015 c
1016 c                                          /* always behind the read...thus don't
1017 c                                             need to test to see if read sector =
1018 c                                             write sector. */
1019 c          call write_sector(outdcnt,bufptr);
1020 c          outdcnt = outdcnt + nfcbs1;
1021 c          nout = 0;
1022 c          lastsect = lastsect + 1;
1023 c
1024 c    end flush_write;
1025 b
1026 c    fill2: proc(bufptr,nout);
1027 c
1028 c          declare (i,j,nout)      bin fixed(7),
1029 c                  bufptr          pointer,
1030 c                  1 buffer(0:127) based(bufptr),
1031 c                   2 user         bit(8),
1032 c                   2 rest(31)     bit(8);
1033 c
1034 d          do i = nout to nfcbs;
1035 d                  buffer(i).user = 'E5'b4;
1036 e                  do j = 1 to 31;
1037 e                         buffer(i).rest(j) = '00000000'b;
1038 e                  end;
1039 d           end;
1040 c
1041 c          lastdcnt = lastdcnt - 1;
1042 c          lasti = nout - 1;
1043 c          call flush_write(nout,bufptr);
1044 c
1045 d          do i = 0 to nfcbs;                      /* prepare empty sector */
1046 d                  buffer(i).user = 'E5'b4;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1047 e                  do j = 1 to 31;
1048 e                          buffer(i).rest(j) = '00000000'b;
1049 e                  end;
1050 d          end;
1051 c
1052 c                                                  /* clear rest of directory */
1053 d          do while (outdcnt < dcnt);
1054 d                  call write_sector(outdcnt,bufptr);
1055 d                  outdcnt = outdcnt + nfcbs1;
1056 d          end;
1057 c
1058 c      end fill2;
1059 b
1060 c      restore: procedure;
1061 c
1062 c              dphp = seldsk(curdisk);         /* restore drive */
1063 c              call reset();                   /* reset disk system */
1064 c              errorcode = select(curdisk);
1065 c
1066 c              call reboot;
1067 c
1068 c      end restore;
1069 b
1070 b                                       /* read logical record # to dma address */
1071 c      read_sector: procedure(lrcd,dmaaddr);
1072 c              dcl
1073 c                 lrcd      bin fixed(15),
1074 c                 prcd      decimal(7,0),
1075 c                 dmaaddr   pointer;                /* dma address */
1076 c
1077 c              prcd = lrcd/nfcbs1;
1078 c              gtrk = track(prcd);
1079 c              call settrk(gtrk);
1080 c              gsec = sector(prcd);
1081 c              call setsec(gsec);
1082 c
1083 c              call bstdma(dmaaddr);
1084 c              if rdsec() ^= 0 then signal error(71);
1085 c
1086 c      end read_sector;
1087 b
1088 b
1089 b                                       /* write logical record # from dma address */
1090 c      write_sector: procedure(lrcd,dmaaddr);
1091 c              dcl
1092 c                 lrcd         bin fixed(15),
1093 c                 dmaaddr      pointer,             /* dma address */
1094 c                 prcd         decimal(7,0);
1095 c
1096 c              prcd = lrcd/nfcbs1;                   /* #fcbs/phys rec */
1097 c              gtrk = track(prcd);
1098 c              call settrk(gtrk);
1099 c              gsec = sector(prcd);
1100 c              call setsec(gsec);
1101 c
1102 c              call bstdma(dmaaddr);
1103 c              if wrsec(1) ^= 0 then signal error(91);
1104 c
1105 c      end write_sector;
1106 b
1107 b
1108 b                                       /* select disk drive */
1109 c      dselect: procedure((d));
1110 c              dcl
1111 c                  p              ptr,
1112 c                  wdalv(16)      fixed(15) based(p),
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
1113 c                btalv(16)      fixed(7)  based(p),
1114 c                all            bit(16),
1115 c                d              fixed(7);
1116 c
1117 c
1118 c          dcl
1119 c                 1 dpb based (dpbp),
1120 c                   2 sec bit(16),
1121 c                   2 bsh bit(8),
1122 c                   2 blm bit(8),
1123 c                   2 exm bit(8),
1124 c                   2 dsm bit(16),
1125 c                   2 drm bit(16),
1126 c                   2 al0 bit(8),
1127 c                   2 al1 bit(8),
1128 c                   2 cks bit(16),
1129 c                   2 off bit(8);
1130 c
1131 c          if d = 0 then d = curdsk();
1132 c          else d = d - 1;
1133 c
1134 c          errorcode = select(d);                  /* sync BIOS & BDOS */
1135 c          dphp = seldsk(d);
1136 c          if dphp = null then call errprint(errBIOS);/* can't select disk */
1137 c
1138 c          xlt = xlt1;
1139 c          dpbp = dpbptr;
1140 c
1141 c          dspt = decimal(spt/(phymsk + 1));
1142 c          dblk = decimal(conv(blkmsk) + 1);
1143 c                                          /* get directory blocks */
1144 c          p = addr(dir_blks(1));
1145 c          all = al0;
1146 c          substr(all,9) = al1;
1147 c
1148 d          do d = 1 to 16;
1149 d              wdalv(d) = 0;       /* clears dir_blks to 0s */
1150 d              if substr(all,d,1) then
1151 d                  if dsksiz < 255 then
1152 d                      btalv(d) = d - 1;
1153 d                  else
1154 d                      wdalv(d) = d - 1;
1155 d          end;
1156 c
1157 c     end dselect;
1158 b
1159 b
1160 b                                  /* convert logical rcd # to physical sector */
1161 c     sector: procedure(i) returns(fixed(15));
1162 c          dcl
1163 c              i    decimal(7,0);
1164 c
1165 c          return(sectrn(binary(mod(i,dspt),15),xlt));
1166 c
1167 c     end sector;
1168 b
1169 b
1170 b                                  /* logical record # to physical track */
1171 c     track: procedure(i) returns(fixed(15));
1172 c          dcl
1173 c              i    decimal(7,0);
1174 c
1175 c          return(offset + binary(i/dspt,15));
1176 c
1177 c     end track;
1178 b
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1179 b
1180 b                                  /* logical record # to physical block */
1181 c      block: procedure(i) returns(fixed(15));
1182 c              dcl
1183 c                  i decimal(7,0);
1184 c
1185 c              return(binary(i/dblk,15));
1186 c
1187 c      end block;
1188 b
1189 b                                  /* block to logical sector */
1190 c      bsec: procedure(i) returns(decimal(7,0));
1191 c              dcl
1192 c                  i     fixed(15);
1193 c
1194 c              if i > dsksiz then signal error(83);
1195 c
1196 c              return(decimal(i) * dblk);
1197 c
1198 c      end bsec;
1199 b
1200 b                              /* convert fixed(7) to fixed(15) w/o sign extension */
1201 c      conv: procedure(i) returns(fixed(15));
1202 c              dcl
1203 c                  i       fixed(7),
1204 c                  j       fixed(15),
1205 c                  p       ptr,
1206 c                  n       fixed(7) based(p);
1207 c
1208 c              p = addr(j);
1209 c              j = 0;
1210 c              n = i;
1211 c              return(j);
1212 c      end conv;
1213 b
1214 b                                  /* test for console break */
1215 c      break_test: procedure ext;
1216 c
1217 c              if con_break() then signal error(85);
1218 c
1219 c      end break_test;
1220 b
1221 b
1222 b                                  /* test for console break */
1223 c      con_break: procedure returns(bit(1));
1224 c              dcl
1225 c                  c char(1);
1226 c
1227 d              if break() then do;
1228 d                  c = rdcon();
1229 d                  if c ^= '^S' then return(TRUE);
1230 d              end;
1231 c              return(FALSE);
1232 c
1233 c      end con_break;
1234 b
1235 b      end initdir;
   NO ERROR(S) IN PASS 1

   NO ERROR(S) IN PASS 2


CODE SIZE = 181B
DATA AREA = 3710
FREE SYMS = 247F
END  COMPILATION
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        name    'DIOMOD'
                        title   'Direct CP/M Calls From PL/I-80'
                ;
                ;***************************************************************
                ;*                                                             *
                ;*      cp/m calls from pl/i for direct i/o                    *
                ;*                                                             *
                ;***************************************************************
                        public  memptr  ;return pointer to base of free mem
                        public  memsiz  ;return size of memory in bytes
                        public  memwds  ;return size of memory in words
                        public  dfcb0   ;return address of default fcb 0
                        public  dfcb1   ;return address of default fcb 1
                        public  dbuff   ;return address of default buffer
                        public  reboot  ;system reboot (#0)
                        public  rdcon   ;read console character (#1)
                        public  wrcon   ;write console character(#2)
                        public  rdrdr   ;read reader character (#3)
                        public  wrpun   ;write punch character (#4)
                        public  wrlst   ;write list character (#5)
                        public  coninp  ;direct console input (#6a)
                        public  conout  ;direct console output (#6b)
                        public  rdstat  ;read console status (#6c)
                        public  getio   ;get io byte (#8)
                        public  setio   ;set i/o byte (#9)
                        public  wrstr   ;write string (#10)
                        public  rdbuf   ;read console buffer (#10)
                        public  break   ;get console status (#11)
                        public  vers    ;get version number (#12)
                        public  reset   ;reset disk system (#13)
                        public  select  ;select disk (#14)
                        public  open    ;open file (#15)
                        public  close   ;close file (#16)
                        public  sear    ;search for file (#17)
                        public  searn   ;search for next (#18)
                        public  delete  ;delete file (#19)
                        public  rdseq   ;read file sequential mode (#20)
                        public  wrseq   ;write file sequential mode (#21)
                        public  make    ;create file (#22)
                        public  rename  ;rename file (#23)
                        public  logvec  ;return login vector (#24)
                        public  curdsk  ;return current disk number (#25)
                        public  setdma  ;set DMA address (#26)
                        public  allvec  ;return address of alloc vector (#27)
                        public  wpdisk  ;write protect disk (#28)
                        public  rovec   ;return read/only vector (#29)
                        public  filatt  ;set file attributes (#30)
                        public  getdpb  ;get base of disk parm block (#31)
                        public  getusr  ;get user code (#32a)
                        public  setusr  ;set user code (#32b)
                        public  rdran   ;read random (#33)
                        public  wrran   ;write random (#34)
                        public  filsiz  ;random file size (#35)
                        public  setrec  ;set random record pos (#36)
                        public  resdrv  ;reset drive (#37)
                        public  wrranz  ;write random, zero fill (#40)
```

CP/M plus Ver 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
                        public  sgscb   ;set/get System Control Block byte/word
                ;
                ;
                        extrn   ?begin  ;beginning of free list
                        extrn   ?boot   ;system reboot entry point
                        extrn   ?bdos   ;bdos entry point
                        extrn   ?dfcb0  ;default fcb 0
                        extrn   ?dfcb1  ;default fcb 1
                        extrn   ?dbuff  ;default buffer
                ;
                ;***************************************************************
                ;*                                                             *
                ;*       equates for interface to cp/m bdos                    *
                ;*                                                             *
                ;***************************************************************
000D =          cr      equ     0dh     ;carriage return
000A =          lf      equ     0ah     ;line feed
001A =          eof     equ     1ah     ;end of file
                ;
0001 =          readc   equ     1       ;read character from console
0002 =          writc   equ     2       ;write console character
0003 =          rdrf    equ     3       ;reader input
0004 =          punf    equ     4       ;punch output
0005 =          listf   equ     5       ;list output function
0006 =          diof    equ     6       ;direct i/o, version 2.0
0007 =          getiof  equ     7       ;get i/o byte
0008 =          setiof  equ     8       ;set i/o byte
0009 =          printf  equ     9       ;print string function
000A =          rdconf  equ     10      ;read console buffer
000B =          statf   equ     11      ;return console status
000C =          versf   equ     12      ;get version number
000D =          resetf  equ     13      ;system reset
000E =          seldf   equ     14      ;select disk function
000F =          openf   equ     15      ;open file function
0010 =          closef  equ     16      ;close file
0011 =          serchf  equ     17      ;search for file
0012 =          serchn  equ     18      ;search next
0013 =          deletf  equ     19      ;delete file
0014 =          readf   equ     20      ;read next record
0015 =          writf   equ     21      ;write next record
0016 =          makef   equ     22      ;make file
0017 =          renamf  equ     23      ;rename file
0018 =          loginf  equ     24      ;get login vector
0019 =          cdiskf  equ     25      ;get current disk number
001A =          setdmf  equ     26      ;set dma function
001B =          getalf  equ     27      ;get allocation base
001C =          wrprof  equ     28      ;write protect disk
001D =          getrof  equ     29      ;get r/o vector
001E =          setatf  equ     30      ;set file attributes
001F =          getdpf  equ     31      ;get disk parameter block
0020 =          userf   equ     32      ;set/get user code
0021 =          rdranf  equ     33      ;read random
0022 =          wrranf  equ     34      ;write random
0023 =          filszf  equ     35      ;compute file size
0024 =          setrcf  equ     36      ;set random record position
0025 =          rsdrvf  equ     37      ;reset drive function
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
002B =          wrrnzf  equ     40      ;write random zero fill
0031 =          scbf    equ     49      ;set/get SCB
                ;
                ;       utility functions
                ;*****************************************************************
                ;*                                                               *
                ;*       general purpose routines used upon entry                *
                ;*                                                               *
                ;*****************************************************************
                ;
                getp1:  ;get single byte parameter to register e
0000 5E                 mov     e,m             ;low (addr)
0001 23                 inx     h
0002 56                 mov     d,m             ;high(addr)
0003 EB                 xchg                    ;hl = .char
0004 5E                 mov     e,m             ;to register e
0005 C9                 ret
                ;
                getp2:  ;get single word value to DE
                getp2i: ;(equivalent to getp2)
0006 CD0000             call    getp1
0009 23                 inx     h
000A 56                 mov     d,m             ;get high byte as well
000B C9                 ret
                ;
                getver: ;get cp/m or mp/m version number
000C E5                 push    h               ;save possible data adr
000D 0E0C               mvi     c,versf
000F CD0000             call    ?bdos
0012 E1                 pop     h               ;recall data addr
0013 C9                 ret
                ;
                chkv20: ;check for version 2.0 or greater
0014 CD0C00             call    getver
0017 FE14               cpi     20
0019 D0                 rnc                     ;return if > 2.0
                ;       error message and stop
001A C32300             jmp     vererr          ;version error
                ;
                chkv22: ;check for version 2.2 or greater
001D CD0C00             call    getver
0020 FE22               cpi     22h
0022 D0                 rnc                     ;return if >= 2.2
                vererr:
                        ;version error, report and terminate
0023 112E00             lxi     d,vermsg
0026 0E09               mvi     c,printf
0028 CD0000             call    ?bdos           ;write message
002B C30000             jmp     ?boot           ;and reboot
002E 0D0A4C6174vermsg:  db      cr,lf,'Later CP/M or MP/M Version Required$'
                ;
                ;*****************************************************************
                ;*                                                               *
                ;*****************************************************************
                memptr: ;return pointer to base of free storage
0054 2A0000             lhld    ?begin
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        ret
                ;
                ;*************************************************************
                ;*                                                           *
                ;*************************************************************
                memsiz: ;return size of free memory in bytes
0058 2A0100             lhld    ?bdos+1         ;base of bdos
005B EB                 xchg                    ;de = .bdos
005C 2A0000             lhld    ?begin          ;beginning of free storage
005F 7B                 mov     a,e             ;low(.bdos)
0060 95                 sub     l               ;-low(begin)
0061 6F                 mov     l,a             ;back to l
0062 7A                 mov     a,d             ;high(.bdos)
0063 9C                 sbb     h
0064 67                 mov     h,a             ;hl = mem size remaining
0065 C9                 ret
                ;
                ;*************************************************************
                ;*                                                           *
                ;*************************************************************
                memwds: ;return size of free memory in words
0066 CD5800             call    memsiz          ;hl = size in bytes
0069 7C                 mov     a,h             ;high(size)
006A B7                 ora     a               ;cy = 0
006B 1F                 rar                     ;cy = ls bit
006C 67                 mov     h,a             ;back to h
006D 7D                 mov     a,l             ;low(size)
006E 1F                 rar                     ;include ls bit
006F 6F                 mov     l,a             ;back to l
0070 C9                 ret                     ;with wds in hl
                ;
                ;*************************************************************
                ;*                                                           *
                ;*************************************************************
                dfcb0:  ;return address of default fcb 0
0071 210000             lxi     h,?dfcb0
0074 C9                 ret
                ;
                ;*************************************************************
                ;*                                                           *
                ;*************************************************************
                dfcb1:  ;return address of default fcb 1
0075 210000             lxi     h,?dfcb1
0078 C9                 ret
                ;
                ;*************************************************************
                ;*                                                           *
                ;*************************************************************
                dbuff:  ;return address of de._ult buffer
0079 210000             lxi     h,?dbuff
007C C9                 ret
                ;
                ;*************************************************************
                ;*                                                           *
                ;*************************************************************
                reboot: ;system reboot (#0)
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        jmp     ?boot
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                rdcon:  ;read console character (#1)
                        ;return character value to stack
0080 0E01               mvi     c,readc
0082 C38C00             jmp     chrin           ;common code to read char
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                wrcon:  ;write console character(#2)
                        ;1->char(1)
0085 0E02               mvi     c,writc         ;console write function
0087 C39C00             jmp     chrout          ;to write the character
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                rdrdr:  ;read reader character (#3)
008A 0E03               mvi     c,rdrf          ;reader function
                chrin:
                        ;common code for character input
008C CD0000             call    ?bdos           ;value returned to A
008F E1                 pop     h               ;return address
0090 F5                 push    psw             ;character to stack
0091 33                 inx     sp              ;delete flags
0092 3E01               mvi     a,1             ;character length is 1
0094 E9                 pchl                    ;back to calling routine
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                wrpun:  ;write punch character (#4)
                        ;1->char(1)
0095 0E04               mvi     c,punf          ;punch output function
0097 C39C00             jmp     chrout          ;common code to write chr
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                wrlst:  ;write list character (#5)
                        ;1->char(1)
009A 0E05               mvi     c,listf         ;list output function
                chrout:
                        ;common code to write character
                        ;1-> character to write
009C CD0000             call    getp1           ;output char to register e
009F C30000             jmp     ?bdos           ;to write and return
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                coninp: ;perform console input, char returned in stack
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
00A2 21AE00          lxi   h,chrstr       ;return address
00A5 E5              push  h              ;to stack for return
00A6 2A0100          lhld  ?boot+1        ;base of bios jmp vector
00A9 110600          lxi   d,2*3          ;offset to jmp conin
00AC 19              dad   d
00AD E9              pchl                 ;return to chrstr
                ;
                chrstr: ;create character string, length 1
00AE E1              pop   h              ;recall return address
00AF F5              push  psw            ;save character
00B0 33              inx   sp             ;delete psw
00B1 3E01            mvi   a,1            ;string length is 1
00B3 E9              pchl                 ;return to caller
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                conout: ;direct console output
                        ;1->char(1)
00B4 CD0000          call  getp1          ;get parameter
00B7 4B              mov   c,e            ;character to c
00B8 2A0100          lhld  ?boot+1        ;base of bios jmp
00BB 110900          lxi   d,3*3          ;console output offset
00BE 19              dad   d              ;hl = .jmp conout
00BF E9              pchl                 ;return through handler
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                rdstat: ;direct console status read
00C0 21EE00          lxi   h,rdsret       ;read status return
00C3 E5              push  h              ;return to rdsret
00C4 2A0100          lhld  ?boot+1        ;base of jmp vector
00C7 110300          lxi   d,1*3          ;offset to .jmp const
00CA 19              dad   d              ;hl = .jmp const
00CB E9              pchl
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                getio:  ;get io byte (#8)
00CC 0E07            mvi   c,getiof
00CE C30000          jmp   ?bdos          ;value returned to A
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
                setio:  ;set i/o byte (#9)
                        ;1->i/o byte
00D1 CD0000          call  getp1          ;new i/o byte to E
00D4 0E08            mvi   c,setiof
00D6 C30000          jmp   ?bdos          ;return through bdos
                ;
                ;*****************************************************
                ;*                                                   *
                ;*****************************************************
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        wrstr:  ;write string (#10)
                                ;1->addr(string)
00D9 CD0600                     call    getp2           ;get parameter value to DE
00DC 0E09                       mvi     c,printf        ;print string function
00DE C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        rdbuf:  ;read console buffer (#10)
                                ;1->addr(buff)
00E1 CD0600                     call    getp2i          ;DE = .buff
00E4 0E0A                       mvi     c,rdconf        ;read console function
00E6 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        break:  ;get console status (#11)
00E9 0E0B                       mvi     c,statf
00EB CD0000                     call    ?bdos           ;return through bdos
                        ;
                        rdsret: ;return clean true value
00EE B7                         ora     a               ;zero?
00EF C8                         rz                      ;return if so
00F0 3EFF                       mvi     a,0ffh          ;clean true value
00F2 C9                         ret
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        vers:   ;get version number (#12)
00F3 0E0C                       mvi     c,versf
00F5 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        reset:  ;reset disk system (#13)
00F8 0E0D                       mvi     c,resetf
00FA C30000                     jmp     ?bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        select: ;select disk (#14)
                                ;1->fixed(7) drive number
00FD CD0000                     call    getp1           ;disk number to E
0100 0E0E                       mvi     c,seldf
0102 C30000                     jmp     ?bdos           ;return through bdos
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        open:   ;open file (#15)
                                ;1-> addr(fcb)
0105 CD0600                     call    getp2i          ;fcb address to de
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SFR. #

```
                        ;*************************************************************
                        ;*                                                           *
                        ;*************************************************************
                        close:  ;close file (#16)
                                ;1-> addr(fcb)
0108 0E0F                       mvi     c,openf
010A C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;*************************************************************
                        ;*                                                           *
                        ;*************************************************************
                        close:  ;close file (#16)
                                ;1-> addr(fcb)
010D CD0600                     call    getp2i          ;.fcb to DE
0110 0E10                       mvi     c,closef
0112 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;*************************************************************
                        ;*                                                           *
                        ;*************************************************************
                        sear:   ;search for file (#17)
                                ;1-> addr(fcb)
0115 CD0600                     call    getp2i          ;.fcb to DE
0118 0E11                       mvi     c,serchf
011A C30000                     jmp     ?bdos
                        ;
                        ;*************************************************************
                        ;*                                                           *
                        ;*************************************************************
                        searn:  ;search for next (#18)
011D 0E12                       mvi     c,serchn        ;search next function
011F C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;*************************************************************
                        ;*                                                           *
                        ;*************************************************************
                        delete: ;delete file (#19)
                                ;1-> addr(fcb)
0122 CD0600                     call    getp2i          ;.fcb to DE
0125 0E13                       mvi     c,deletf
0127 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;*************************************************************
                        ;*                                                           *
                        ;*************************************************************
                        rdseq:  ;read file sequential mode (#20)
                                ;1-> addr(fcb)
012A CD0600                     call    getp2i          ;.fcb to DE
012D 0E14                       mvi     c,readf
012F C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;*************************************************************
                        ;*                                                           *
                        ;*************************************************************
                        wrseq:  ;write file sequential mode (#21)
                                ;1-> addr(fcb)
0132 CD0600                     call    getp2i          ;.fcb to DE
0135 0E15                       mvi     c,writf
0137 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;*************************************************************
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        ;*                                                    *
                        ;******************************************************
                        make:   ;create file (#22)
                                ;1-> addr(fcb)
013A CD0600                     call    getp2i          ;.fcb to DE
013D 0E16                       mvi     c,makef
013F C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        rename: ;rename file (#23)
                                ;1-> addr(fcb)
0142 CD0600                     call    getp2i          ;.fcb to DE
0145 0E17                       mvi     c,renamf
0147 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        logvec: ;return login vector (#24)
014A 0E18                       mvi     c,loginf
014C C30000                     jmp     ?bdos           ;return through BDOS
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        curdsk: ;return current disk number (#25)
014F 0E19                       mvi     c,cdiskf
0151 C30000                     jmp     ?bdos           ;return value in A
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        setdma: ;set DMA address (#26)
                                ;1-> pointer (dma address)
0154 CD0600                     call    getp2           ;dma address to DE
0157 0E1A                       mvi     c,setdmf
0159 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        allvec: ;return address of allocation vector (#27)
015C 0E1B                       mvi     c,getalf
015E C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        wpdisk: ;write protect disk (#28)
0161 CD1400                     call    chkv20          ;must be 2.0 or greater
0164 0E1C                       mvi     c,wrprof
0166 C30000                     jmp     ?bdos
                        ;
                        ;******************************************************
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```
                        ;*
                        ;**************************************************************
                        rovec:  ;return read/only vector (#29)
0169 CD1400                     call    chkv20          ;must be 2.0 or greater
016C 0E1D                       mvi     c,getrof
016E C30000                     jmp     ?bdos           ;value returned in HL
                        ;
                        ;**************************************************************
                        ;*                                                            *
                        ;**************************************************************
                        filatt: ;set file attributes (#30)
                                ;1-> addr(fcb)
0171 CD1400                     call    chkv20          ;must be 2.0 or greater
0174 CD0600                     call    getp2i          ;.fcb to DE
0177 0E1E                       mvi     c,setatf
0179 C30000                     jmp     ?bdos
                        ;
                        ;**************************************************************
                        ;*                                                            *
                        ;**************************************************************
                        getdpb: ;get base of current disk parm block (#31)
017C CD1400                     call    chkv20          ;check for 2.0 or greater
017F 0E1F                       mvi     c,getdpf
0181 C30000                     jmp     ?bdos           ;addr returned in HL
                        ;
                        ;**************************************************************
                        ;*                                                            *
                        ;**************************************************************
                        getusr: ;get user code to register A
0184 CD1400                     call    chkv20          ;check for 2.0 or greater
0187 1EFF                       mvi     e,0ffh          ;to get user code
0189 0E20                       mvi     c,userf
018B C30000                     jmp     ?bdos
                        ;
                        ;**************************************************************
                        ;*                                                            *
                        ;**************************************************************
                        setusr: ;set user code
018E CD1400                     call    chkv20          ;check for 2.0 or greater
0191 CD0000                     call    getp1           ;code to E
0194 0E20                       mvi     c,userf
0196 C30000                     jmp     ?bdos
                        ;
                        ;**************************************************************
                        ;*                                                            *
                        ;**************************************************************
                        rdran:  ;read random (#33)
                                ;1-> addr(fcb)
0199 CD1400                     call    chkv20          ;check for 2.0 or greater
019C CD0600                     call    getp2i          ;.fcb to DE
019F 0E21                       mvi     c,rdranf
01A1 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;**************************************************************
                        ;*                                                            *
                        ;**************************************************************
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        wrran:  ;write random (#34)
                                ;1-> addr(fcb)
01A4 CD1400                     call    chkv20          ;check for 2.0 or greater
01A7 CD0600                     call    getp2i          ;.fcb to DE
01AA 0E22                       mvi     c,wrranf
01AC C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        filsiz: ;compute file size (#35)
01AF CD1400                     call    chkv20          ;must be 2.0 or greater
01B2 CD0600                     call    getp2           ;.fcb to DE
01B5 0E23                       mvi     c,filszf
01B7 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        setrec: ;set random record position (#36)
01BA CD1400                     call    chkv20          ;must be 2.0 or greater
01BD CD0600                     call    getp2           ;.fcb to DE
01C0 0E24                       mvi     c,setrcf
01C2 C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        resdrv: ;reset drive function (#37)
                                ;1->drive vector - bit(16)
01C5 CD1D00                     call    chkv22          ;must be 2.2 or greater
01C8 CD0600                     call    getp2           ;drive reset vector to DE
01CB 0E25                       mvi     c,rsdrvf
01CD C30000                     jmp     ?bdos           ;return through bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        wrranz: ;write random, zero fill function
                                ;1-> addr(fcb)
01D0 CD1D00                     call    chkv22  ;must be 2.2 or greater
01D3 CD0600                     call    getp2i          ;.fcb to DE
01D6 0E28                       mvi     c,wrrnzf
01D8 C30000                     jmp     ?bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
                        ;***************************************************************
                        sgscb:  ;set/get SCB byte/word
                               ;1-> addr(SCB structure)
01DB CD0600                     call    getp2
01DE 0E31                       mvi     c,scbf
01E0 C30000                     jmp     ?bdos
                        ;
                        ;***************************************************************
                        ;*                                                             *
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```
617                     ;*****************************************************
618  01E3                       end
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

ALLVEC	015C	44	492#									
BREAK	00E9	28	355#									
CDISKF	0019	100#	477									
CHKV20	0014	145#	500	508	517	526	534	543	553	563	572	
		581										
CHKV22	001D	152#	591	601								
CHRIN	008C	233	248#									
CHROUT	009C	241	263	271#								
CHRSTR	00AE	281	288#									
CLOSE	010D	33	399#									
CLOSEF	0010	91#	402									
CONINP	00A2	21	280#									
CONOUT	00B4	22	298#									
CR	000D	72#	162									
CURDSK	014F	42	476#									
DBUFF	0079	14	217#									
DELETE	0122	36	424#									
DELETF	0013	94#	427									
DFCB0	0071	12	203#									
DFCB1	0075	13	210#									
DIOF	0006	81#										
EOF	001A	74#										
FILATT	0171	47	515#									
FILSIZ	01AF	53	571#									
FILSZF	0023	110#	574									
GETALF	001B	102#	493									
GETDPB	017C	48	525#									
GETDPF	001F	106#	527									
GETIO	00CC	24	321#									
GETIOF	0007	82#	322									
GETP1	0000	123#	133	274	300	330	384	544				
GETP2	0006	131#	339	485	573	582	592	611				
GETP2I	0006	132#	348	392	401	410	426	435	444	453	462	
		518	554	564	602							
GETROF	001D	104#	509									
GETUSR	0184	49	533#									
GETVER	000C	138#	146	153								
LF	000A	73#	162									
LISTF	0005	80#	270									
LOGINF	0018	99#	470									
LOGVEC	014A	41	469#									
MAKE	013A	39	451#									
MAKEF	0016	97#	454									
MEMPTR	0054	9	167#									
MEMSIZ	0058	10	174#	190								
MEMWDS	0066	11	189#									
OPEN	0105	32	390#									
OPENF	000F	90#	393									
PRINTF	0009	84#	159	340								
PUNF	0004	79#	262									
RDBUF	00E1	27	346#									
RDCON	0080	16	230#									
RDCONF	000A	85#	349									
RDRAN	0199	51	551#									
RDRANF	0021	108#	555									
RDRDR	008A	18	246#									

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

R[illegible]	0003	78#	247	
RDSEQ	012A	37	433#	
RDSRET	00EE	311	359#	
RDSTAT	00C0	23	310#	
READC	0001	76#	232	
READF	0014	95#	436	
REBOOT	007D	15	224#	
RENAME	0142	40	460#	
RENAMF	0017	98#	463	
RFSDRV	01C5	55	589#	
RESET	00F8	30	375#	
RESETF	000D	88#	376	
RDVEC	0169	46	507#	
RSDRVF	0025	112#	593	
SCBF	0031	114#	612	
SEAR	0115	34	408#	
SE[illegible]RN	011D	35	417#	
SELDF	000E	89#	385	
SELECT	00FD	31	382#	
SERCHF	0011	92#	411	
SERCHN	0012	93#	418	
SETATF	001E	105#	519	
SETDMA	0154	43	483#	
SETDMF	001A	101#	486	
SETIO	00D1	25	328#	
SETIOF	0008	83#	331	
SETRCF	0024	111#	583	
SETREC	01BA	54	580#	
SETUSR	018E	50	542#	
SGSCB	01DB	57	609#	
STATF	000B	86#	356	
USERF	0020	107#	536	545
VERERR	0023	150	156#	
VERMSG	002E	158	162#	
VERS	00F3	29	368#	
VERSF	000C	87#	140	369
WPDISK	0161	45	499#	
WRCON	0085	17	238#	
WRITC	0002	77#	240	
WRITF	0015	96#	445	
WRLST	009A	20	268#	
WRPROF	001C	103#	501	
WRPUN	0095	19	260#	
WRRAN	01A4	52	561#	
WRRANF	0022	109#	565	
WRRANZ	01D0	56	599#	
WRRNZF	0028	113#	603	
WRSEQ	0132	38	442#	
WRSTR	00D9	26	337#	
?BDOS	0000	62	141	160 175 250 275 323 332 341 350
		357	370	377 386 394 403 412 419 428 437
		446	455	464 471 478 487 494 502 510 520
		528	537	546 556 566 575 584 594 604 613
?BEGIN	0000	60	168	177
?BOOT	0000	61	161	225 283 302 313
?DBUFF	0000	65	218	

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

?DFCB0	0000	63	204
?DFCB1	0000	64	211

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

```
                        name    'BIOSMOD'
                        title   'Direct BIOS Calls From PL/I-80 for CP/M 3.0'
                ;
                ;***************************************************************
                ;*                                                             *
                ;*      bios calls from pl/i for track, sector io              *
                ;*                                                             *
                ;***************************************************************
                        public  settrk  ;set track number
                        public  setsec  ;set sector number
                        public  rdsec   ;read sector
                        public  wrsec   ;write sector
                        public  seldsk  ;select disk & return the addr(DPH)
                        public  sectrn  ;translate sector # given translate table
                        public  bstdma  ;set dma
                ;
                ;
                        extrn   ?boot   ;system reboot entry point
                        extrn   ?bdos   ;bdos entry point
                ;
                ;       utility functions
                ;
                ;***************************************************************
                ;***************************************************************
                ;*                                                             *
                ;*       general purpose routines used upon entry              *
                ;*                                                             *
                ;***************************************************************
                ;
                ;
                getp2:  ;get single word value to DE
0000 5E                 mov     e,m
0001 23                 inx     h
0002 56                 mov     d,m
0003 23                 inx     h
0004 E5                 push    h
0005 EB                 xchg
0006 5E                 mov     e,m
0007 23                 inx     h
0008 56                 mov     d,m
0009 E1                 pop     h
000A C9                 ret
                ;
                ;
                ;***************************************************************
                ;*                                                             *
                ;***************************************************************
                settrk: ;set track number 0-76, 0-65535 in BC
                        ;1-> track #
000B CD0000             call getp2
000E EB                 xchg
000F 226D00             shld    BCREG
0012 3E0A               mvi     a,0ah
0014 C35D00             jmp gobios
                ;
                ;***************************************************************
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        ;*                                                    *
                        ;******************************************************
                        setsec: ;set sector number 1 - sectors per track
                                ;1-> sector #
0017 CD0000                     call getp2
001A EB                         xchg
001B 226D00                     shld    BCREG
001E 3E0B                       mvi     a,0bh
0020 C35D00                     jmp gobios
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        rdsec:  ;read current sector into sector at dma addr
                                ;returns        0 if no errors
                                ;               1 non-recoverable error
0023 3E0D                       mvi     a,0dh
0025 C35D00                     jmp gobios
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        wrsec:  ;writes contents of sector at dma addr to current sector
                                ;returns        0 errors occured
                                ;               1 non-recoverable error
0028 CD0000                     call getp2
002B EB                         xchg
002C 226D00                     shld    BCREG
002F 3E0E                       mvi     a,0eh
0031 C35D00                     jmp gobios
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        ;
                        seldsk: ; selects disk

0034 CD0000                     call getp2
0037 7B                         mov     a,e
0038 326D00                     sta     BCREG
003B 3E09                       mvi     a,9
003D C35D00                     jmp     gobios
                        ;
                        ;******************************************************
                        ;*                                                    *
                        ;******************************************************
                        ;
                        sectrn: ;translate sector #
0040 CD0000                     call    getp2
0043 EB                         xchg
0044 226D00                     shld    BCREG
0047 EB                         xchg
0048 CD0000                     call    getp2
004B EB                         xchg
004C 226F00                     shld    DEREG
004F 3E10                       mvi     a,10h
0051 C35D00                     jmp     gobios
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        ;
                        bstdma: ;set dma
0054 CD0000             call    getp2
0057 EB                 xchg
0058 226D00             shld    BCREG
005B 3E0C               mvi     a,0ch
                        ;       jmp     gobios
                        ;
                        ;***************************************************************
                        ;***************************************************************
                        ;***************************************************************
                        ;*                                                             *
                        ;*       call BDOS                                             *
                        ;*                                                             *
                        ;***************************************************************
                        ;
                        ;
                        gobios:
005D 326B00             sta     FUNC    ;load BIOS function #
0060 216B00             lxi     h,FUNC
0063 EB                 xchg            ; address of BIOSPB in DE
0064 0E32               mvi     c,032h  ; BDOS function 50 call
0066 C30000             jmp     ?bdos
                        ;
                        ;
0069 6B00               BIOSPB:         dw      FUNC
006B 00                 FUNC:           db      0
006C 00                 AREG:           db      0
006D 0000               BCREG:          dw      0
006F 0000               DEREG:          dw      0
0071 0000               HLREG:          dw      0
                        ;
0073                            end
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
AREG            006C  140#
BCREG           006D   52    63    83    95   106   117   141#
BIOSPB          0069  138#
BSTDMA          0054   15   114#
DEREG           006F  110   142#
FUNC            006B  131   132   138   139#
GETP2           0000   31#   50    61    81    93   104   108   115
GOBIOS          005D   54    65    74    85    97   112   130#
HLREG           0071  143#
RDSEC           0023   11    70#
SECTRN          0040   14   103#
SELDSK          0034   13    91#
SETSEC          0017   10    59#
SETTRK          000B    9    48#
WRSEC           0028   12    78#
?BDOS           0000   19   135
?BOOT           0000   18
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____